Lab 7: Inheritance and Polymorphism

Objectives

- To understand the concepts of inheritance and polymorphism.
- To be able develop a subclass from a superclass through inheritance.
- To understand the concepts of dynamic binding, and generic programming.
- To be able restrict access to data and methods using the *protected* visibility modifier.

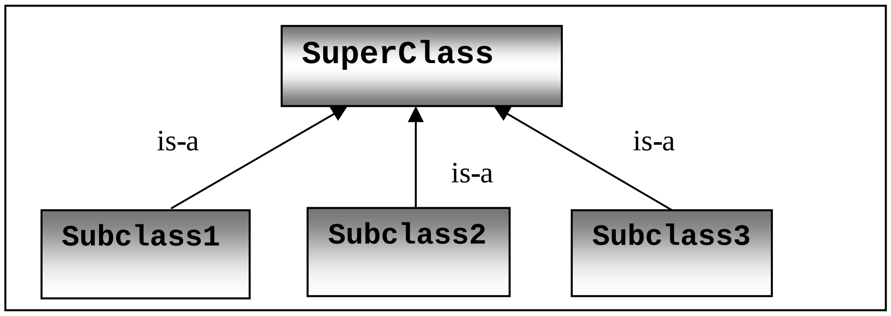
Definition

Inheritance: allows you to derive new classes from existing classes. Inheritance represents the is-a relationship.

Polymorphism: is to behave differently with different subclasses.

As objects do not exist by themselves but are instances of a ***class***, a class can inherit the features of another class and add its own modifications. (This could mean restrictions or additions to its functionality). Inheritance aids in the reuse of code.

Classes can have 'Children' that is, one class can be created out of another class. The original or parent class is known as the ***SuperClass*** (or base class). The child class is known as the ***SubClass*** (or derived class).



A **SubClass** inherits all the *attributes* and *behaviors* of the **SuperClass**, and may have additional attributes and behaviors.

Syntax

In Java inheritance is represented using *extends* keyword as follows:

```
[modifiers] super-class-name { //Data members and methods …}

[modifiers] subclass-name extends super-class-name {…}
```

Exercises

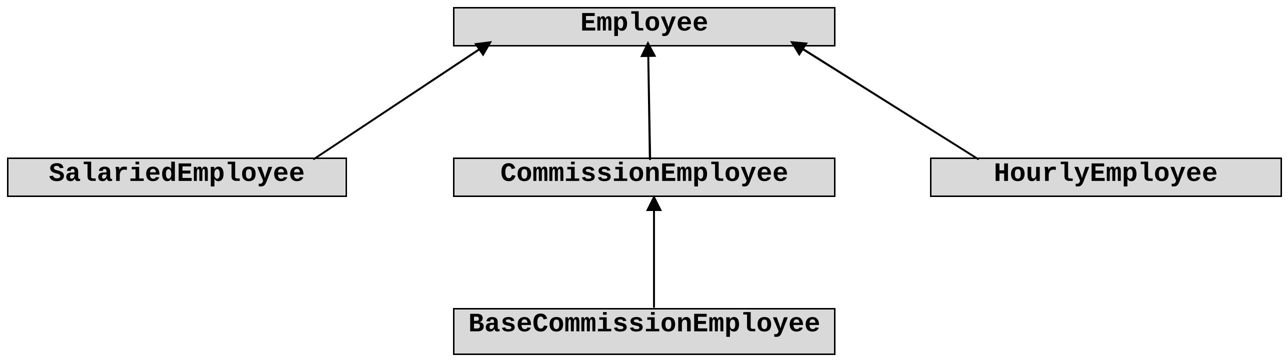
1. Design a class named `Account` that contains:

- An `int` data field named `id` for the account (default `0`).
- A `double` data field named `balance` for the account (default `0`).
- A `double` data field named `annualInterestRate` that stores the current interest rate (default `0`).
- A `Date` data field named `dateCreated` that stores the date when the account was created.
- A no-arg constructor that creates a default account.
- The accessor and mutator methods for `id`, `balance`, and `annualInterestRate`.
- The accessor method for `dateCreated`.
- A method named `getMonthlyInterestRate()` that returns the monthly interest rate.
- A method named `withDraw` that withdraws a specified amount from the account.
- A method named `deposit` that deposits a specified amount to the account.

The `Account` class was created to model a bank account. An account has the properties account number, balance, annual interest rate, and date created, and methods to deposit and withdraw. Create two subclasses for checking and saving accounts. A checking account has an overdraft limit, but a savings account cannot be overdrawn.

Draw the UML diagram for the classes. Implement the classes. Write a test program that creates objects of `Account`, **`SavingsAccount`**, and **`CheckingAccount`**, and invokes their `toString()` methods.

2. Create the classes in the following inheritance hierarchy. An **`Employee`** should have a first name, last name and social security number. In addition the **`SalariedEmployee`** should have a weekly salary; an **`HourlyEmployee`** should have a wage and a number of hours worked; a **`CommissionEmployee`** should have a commission rate and gross sales; and a **`BaseCommissionEmployee`** should have a base salary. Each class should have an appropriate constructor, set and get methods. Each Employee should have an `earning()` and `toString()` methods.
Create a driver class with an array of Employees. Then access the `earning()` and `toString()` methods for each employee. [Now do you understand the concept of **Polymorphism**? What does it mean?]


Employee
SalariedEmployee
CommissionEmployee
HourlyEmployee
BaseCommissionEmployee